Modeling strategies using visual programming in OpenAlea

Christophe Pradal[1], **Thomas Cokelaer**[2], **Daniel Barbeau**[2], **Eric Moscardi**[2], **Jérôme Chopard**[2] **and Christophe Godin**[2]
[1]CIRAD/[2]INRIA, Virtual Plants INRIA Team, UMR DAP, TA A96/02, 34398 Montpellier Cedex 5
email : christophe.pradal@cirad.fr

Keywords : plant modeling, Python, dataflow, modeling software

Introduction

Plant modeling involves a large variety of scientific background and technical knowledge, and scientists need to cope with an increasing amount of data from field measurements. In order to validate hypothesis using functional-structural plant models (FSPM), it becomes essential to share expertise from different groups so as to be able to *(i)* design experiments to analyze heterogeneous data sets, *(ii)* combine independent modules, and *(iii)* perform pre or post processing such as statistical analysis or visualization on simulated processes and observed phenomena. All the modeling knowledge (from data analysis to modeling, simulation and validation) should be gathered and shared within a common framework. OpenAlea provides such a framework through a Visual Programming Environment (VPE) called VisuAlea, a graphical user interface written in Python (Pradal *et al.*, 2008). In this framework, users and modelers are able to dynamically combine existing and independent pieces of software into a customizable dataflow (Johnston *et al.*, 2004).
In this paper, we consider the problem of managing the entire process of plant modeling in VisuAlea. We will describe our attempt to express standard modeling tasks, such as validation of virtual experiments and design patterns to represent feedback loops within a VPE. First, we give a brief overview and status of OpenAlea/VisuAlea. Then, we provide basic dataflow examples to illustrate the potential of VisuAlea. Finally, we discuss advanced topics that enable feedback and parallelism within VisuAlea.

The OpenAlea dataflow and its visual programming environment, VisuAlea

VisuAlea allows to graphically assemble functionalities into an application. These functionalities, abstracted as nodes with well defined inputs and outputs, are connected together through edges that represent the flow of data between them. A graph of such nodes can be encapsulated into a composite node. This composition lets users factorize common processes into a single node to create reusable subsystems. Consequently, a dataflow may be a very complex combination of nodes with different levels of granularity.
VisuAlea allows *(i)* fast design of new models, *(ii)* reuse of existing standard scientific packages, and *(iii)* automatic creation of graphical applications. However, like other VPEs, VisuAlea has major drawbacks compared to textual programming: for instance, it is difficult to represent iteration structures since a dataflow is by definition a directed acyclic graph (Johnston *et al.*, 2004). Therefore, we introduced functional operators like *map* (see the *Advanced techniques* section) that allows to by-pass this limitation. Another solution is to call the dataflow directly from the python language. Consequently, VisuAlea empowers plant scientists to graphically design software pipelines and interactive workflows as illustrated hereafter.

From scientific workflow to coherent pipeline

Plant modeling activities such as plant architecture analysis, 3D plant reconstruction or validation of virtual experiments often require assembling a heterogeneous set of tools and processing data in an automated way. Such assembly consists of several components connected into a pipeline. For instance, model building in plant architecture analysis can be designed as follows (Guédon *et al.*, 2002): *(i)* data exploratory, *(ii)* model specification, *(iii)* model inference, and *(iv)* model validation. VPEs are well adapted to data analysis pipelines where users can interact with the model by changing its parameters, replacing a model family by another one, and visualizing the result. The former point being eased by a large number of visualization packages (e.g. GnuPlot, R, Matplotlib). As an example, a dataflow that shows how a 2D image analysis pipeline can be easily designed in VisuAlea is provided (Fig 1). OpenAlea offers a complete, intuitive and flexible environment to scientists and researchers for solving complex imaging problems.

This example focuses on image processing but can be extended to many aspects of FSPM. Its visual environment makes it possible quickly and easily integrate custom algorithms, with live execution,verification and validation of their models.

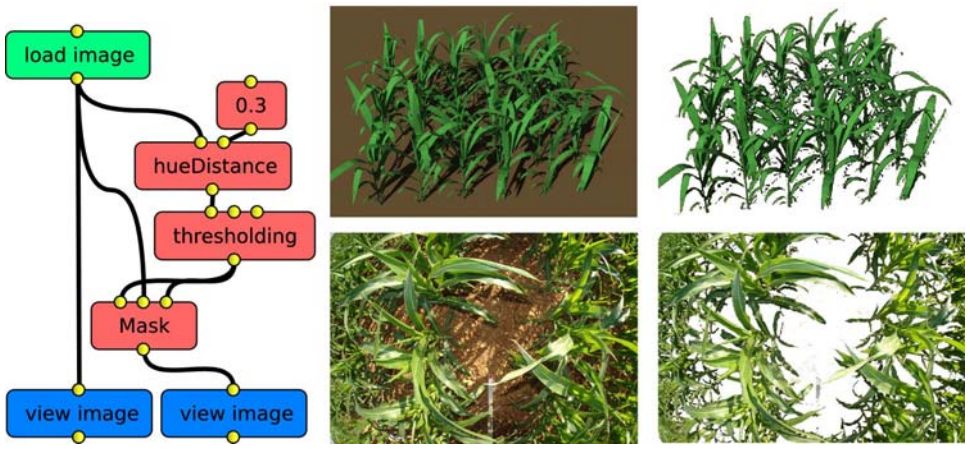

Fig 1: ***A linear pipeline to extract green components*** *– A user has reconstructed a maize field with a dataflow provided by the "Adel" package (Fournier et al., 2003) available in OpenAlea. In order to validate the virtual experiment, he has to compare the reconstruction with raw images from the field by extracting the green color (plant object). First, the images are loaded (middle column), then a distance map is computed (i.e. difference between the green element (0.3) and the hue of each pixel). Finally, a threshold is applied on this distance map. The resulting image is used as a mask on the original data (right column).*

The OpenAlea initiative insists on the reuse of existing components to effectively construct scientific workflows. Simple yet powerful pipelines can be built without knowing any programming language.

Advanced techniques

In FSPM, simulation of plant growth requires to take into account the plant environment and therefore to include feedback loops in the modeling platform and process large data sets. Here, we explain how this can be done.

Conversion of a dataflow into a function – A dataflow is a sequence of operations and like a function it takes input values and returns output values. However, since the input variables are set by the user, it is just a particular instance of a function (*e.g.*, f(3)). Executing the same dataflow with different inputs requires the user to change the parameters manually in the VPE. In order to generalize the dataflow to use it as a function (*e.g.*, f: x→f(x)), a user-set variable has to be replaced by a free variable. In OpenAlea, a free variable is represented by a special node, named "X_i" which transforms a dataflow with fixed variables into a function. A free variable can be shared by several dataflows to represent different functions that share the same variable (*i.e.* f(x)+g(x) as opposed to f(x)+g(y)). Several free variables can be connected to a dataflow to define a function of several variables. Such a dataflow, like any other Python function, can be set as input of a node. Specific nodes (e.g. for, while) allow to evaluate it in loops. This allows for classical *Fibonacci* or factorial functions to be visually implemented.

Applying a dataflow several times – Once a dataflow is defined for a given data set, we usually want to apply it several times on a series of data sets. This is a *map* operation: a function *f* is repeatedly applied on a set $s = \{x_1, x_2,...\}$ of values and yields a new set $s' = \{f(x_1), f(x_2),...\}$.

Parallel processing of large data sets – The *map* operation uses a single processor. To process large data sets, a *parallel map* node is included in OpenAlea. This node is equivalent to the *map* node but distributes the data across multiple processors. This type of parallelism is called SPMD (Single program, multiple data parallelism). We use IPython for parallel computing (Perez *et al.*, 2007): it provides shared-memory or distributed process dispatching strategies.

Feedback loops with scheduling – It is common in FSPMs to have physiological processes with different time scales within a single simulation. In procedural programming, such a case is carried out by nested *for* loops. Visual programming needs an equivalent mechanism. Therefore, a scheduler is required to coordinate the different processes. In OpenAlea, a *scheduler* has been implemented.

For each process, a task is registered; this task is defined by a function (the process) and a frequency of execution. When the dataflow is evaluated, the scheduler controls the execution of each task. The function associated with the task can be either a standard function or built from an already existing dataflow using the *X* node (see above). In VisuAlea, each task is visually separated. Thus, assembly of multiple processes is clear and simple (see Fig 2).

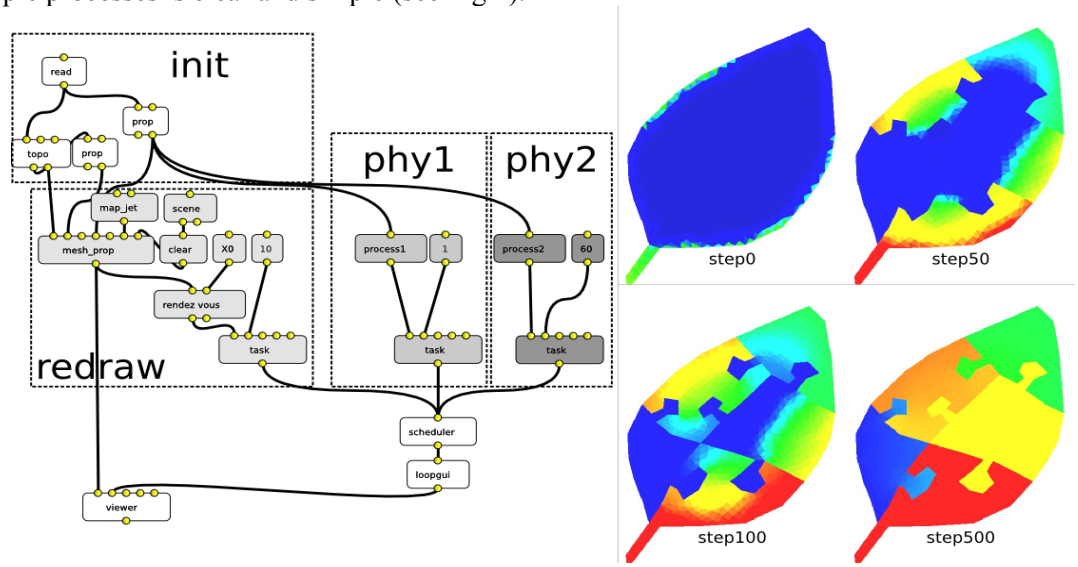

Fig 2: **Simulation of diffusion in a cellular tissue** – The tissue contains 5 cells (represented as jigsaw puzzle elements). A compound is applied along the margins at different rates and diffuses inside the cells but cannot cross their boundaries. Two concurrent physiological processes are run: a fast one (phy1, diffusion of a compound inside the leaf, computed every second) and a slow one (phy2, control of the amount of compound generated along the margins, computed every minute). The current state of the tissue is displayed every ten seconds. The redraw task benefits from an already existing dataflow used to analyze the tissue, combined with an *X* node in order to redraw the tissue when necessary.

Discussion

In this abstract, we have presented new functionalities of OpenAlea/VisuAlea and emphasized the advantages of VPEs and dataflows to manage all aspects of plant modeling: from data analysis and model building to simulation and post-processing.

VisuAlea, like other VPEs, provides a graphical representation of the model that lets the user understand and introspect its structure as well as its inputs and outputs. In VisuAlea, scientific pipelines can be built interactively by reusing components developed and shared by other teams. It enhances communication between modelers, encourages cooperation and facilitates knowledge sharing. Limitations to represent loops visually have been overcome to model feedback between the structure and the function. In addition, advanced techniques such as scheduling and distributed computing have been presented in the context of plant modeling.

Acknowledgments – This project is supported by Agropolis Foundation. The authors thank Christian Fournier for providing maize crop pictures.

References

C. Pradal, S. Dufour-Kowalski, F. Boudon, C. Fournier, and C. Godin. 2008. Openalea: A visual programming and component-based software platform for plant modeling. Functional Plant Biology, 35(10), 751-760.

W.M.. Johnston, J.R. Hanna, and R.J. Millar. 2004. Advances in dataflow programming languages. ACM Comp. Surv. 36(1), 1-34

F. Perez, B. Granger. 2007. IPython: A System for Interactive Scientific Computing. CISE, 21-29

Y. Guédon, D. Barthélémy, Y. Caraglio, E. Costes. 2001. Pattern analysis in branching and axillary flowering sequences. Journal of Theoretical Biology, 212, 481-520

C. Fournier, B. Andrieu, S. Ljutovac and S. Saint-Jean. 2003. ADEL-wheat: a 3D architectural model of wheat development. In: Plant Growth Modeling and Applications (eds Hu & Jaeger), Springer, 54-66.